swiftsimio: A Python library for reading SWIFT data

Josh Borrow[1] **and Alexei Borrisov**[2]

1 Institute for Computational Cosmology, Durham University **2** School of Physics and Astronomy, University of Edinburgh

DOI: 10.21105/joss.02430

Software

- Review
- Repository
- Archive

Editor: Dan Foreman-Mackey

Reviewers:

- @dmentipl
- @mtremmel

Submitted: 26 June 2020
Published: 01 August 2020

Summary

`swiftsimio` is a Python package for reading data created by the SWIFT (Schaller, Gonnet, Chalk, & Draper, 2016) simulation code. SWIFT is designed to run cosmological hydrodynamics simulations that produce petabytes of data, and `swiftsimio` leverages the custom metadata that SWIFT produces to allow for chunked loading of the particle data and to enable integration with `unyt` (Goldbaum, ZuHone, Turk, Kowalik, & Rosen, 2018).

Background

Cosmological galaxy formation simulations are used to track simulated galaxies across cosmic time in an attempt to better understand the process of galaxy formation and evolution. As time has progressed, so has the scale of these simulations. The state-of-the-art simulations planned for the next decade, using codes like SWIFT, will generate petabytes of data thanks to their use of hundreds of billions of particles. Analysing this data presents a unique challenge, with the data reduction performed on either single compute nodes or on individual desktop machines.

In the original EAGLE simulation (Schaye et al., 2015), just the co-ordinates of the gas particles in a single snapshot used 82 Gb of storage space. State-of-the-art simulations are now using at least 10 times as many particles as this, making an analysis pipeline that reads the whole array each time expensive in the best case and infeasible in the worst. There are two useful properties of this data: it is stored in the HDF5 format (The HDF Group, 1997), allowing for easy slicing, and usually users are interested in a very small sub-set of the data, usually less than 1%, at a time.

This requirement to load less than 1% of the data at a time is primarily due to the huge dynamic range present in these simulations (Borrow, Bower, Draper, Gonnet, & Schaller, 2018). Although the simulation volume may be hundreds of megaparsecs on a side, the objects that form under self-gravity are typically less than a few megaparsecs in diameter, representing a very small proportion of the total volume. Users are usually interested in using the particle data present in a few objects (selected using pre-computed data in 'halo catalogues') at any given time.

Structure of a SWIFT snapshot

At pre-determined times during a SWIFT simulation, a full particle dump is performed. This particle dump is stored in the HDF5 format, with arrays corresponding to different particle properties. The overall structure of the file conforms to the Gadget-2 specification (Springel, 2005). This is to enable compatibility with previous analysis pipelines.

The SWIFT snapshot files have main datasets called `PartType{0,1,2,3,4,5}`, each with sub-datasets such as `Coordinates`, `Velocities`, etc. that contain the co-ordinates and velocities for particles of that type respectively. When the file is opened in `swiftsimio`, this is translated to an object hierarchy, so that for a user to access the co-ordinates of the gas particles they would use code similar to:

```
from swiftsimio import load

data = load("/path/to/snapshot.hdf5")
coordinates = data.gas.coordinates
```

where here `coordinates` is an `unyt` array containing the co-ordinates of all of the gas particles in the file with appropriate units and cosmology information attached. This data is loaded lazily, with an array only loaded from file when it is requested for use by the user. This data is then cached for future use.

The table below shows what each particle type corresponds to and how it is accessed in `swiftsimio`.

Particle type	`swiftsimio` name	Description
0	`gas`	Gas particles, the only type of particles to have hydrodynamics calculations performed on them.
1	`dark_matter`	Dark matter particles, only subject to the force of gravity.
2	`boundary`	Boundary particles; the same as dark matter but typically more massive.
3	`second_boundary`	Boundary particles; the same as dark matter but typically more massive.
4	`stars`	Star particles representing either individual stars or a stellar population.
5	`black_holes`	Black hole particles representing individual black holes.

Solving the data reduction challenge

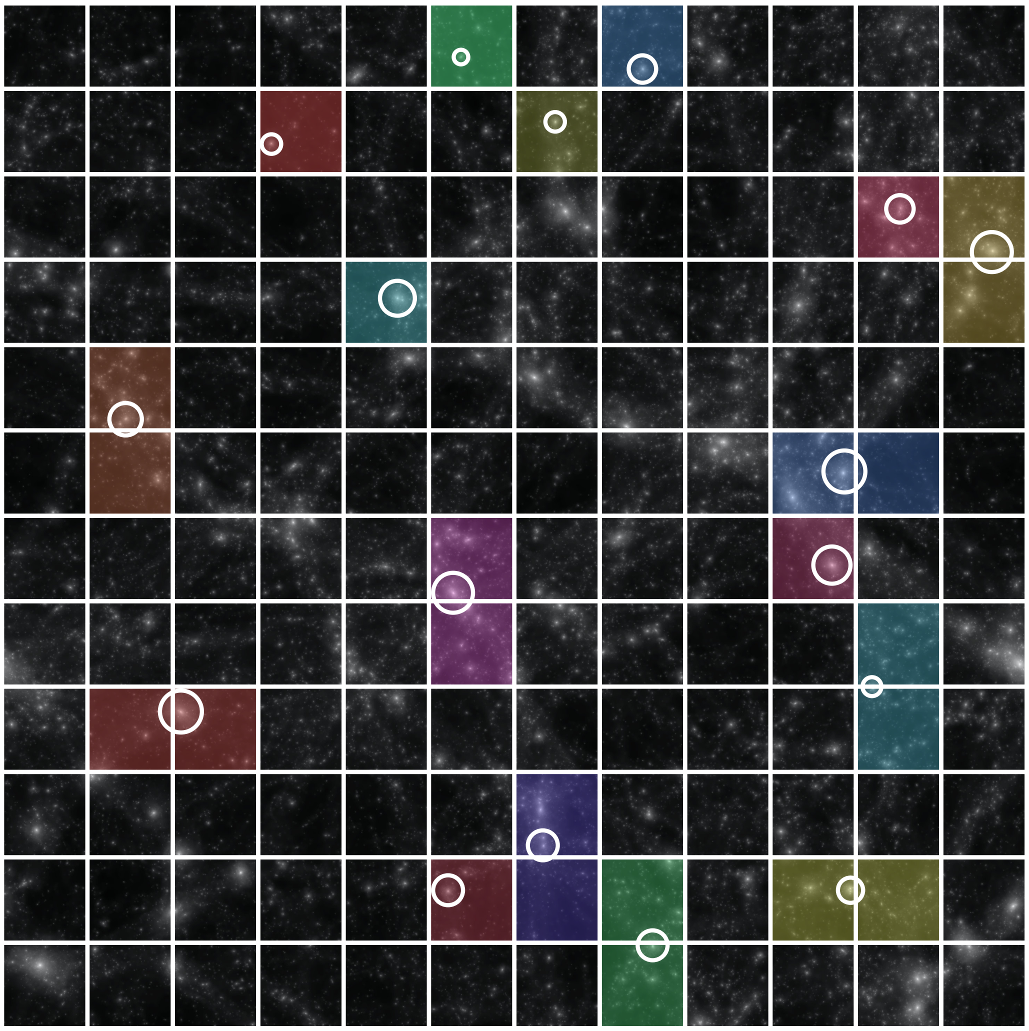

Figure 1: Pictorial representation of the top-level grid in SWIFT. The background shows the distribution of matter in the snapshot, with selected galaxies circled. `swiftsimio` can load the data in the regions that these spheres overlap with, only reading the appropriate particle data from file. Each coloured region shows the top-level cells that would be loaded for the corresponding circled galaxy.

To solve the dynamic range problem, we can turn to the spatial arrangement of the particle data. Simulations in SWIFT are performed in a cuboid volume that is split into a fixed number of top-level cells. When a snapshot of the simulation is dumped, each array is written to disk top-level cell by top-level cell, ensuring a well-characterised order. The order in which the data is stored is then written as metadata in the `Cells` dataset in the snapshot file. The `SWIFTMask` object within `swiftsimio` uses this metadata to provide a mask to be used with the `h5py` library to read a sub-set of the particle data. This process ensures that as little data is read from disk as possible.

The use of this masking functionality is unique to `swiftsimio` and allows for significantly reduced memory footprint and cost relative to reading the full snapshot. Other libraries, such as `yt` (Turk et al., 2011) offer similar functionality through the creation of a hashtable on the first read of the snapshot, but our approach requires no extra computation or files to achieve.

Why swiftsimio?

There are many Python libraries that are able to read the Gadget-style formatted HDF5 data that SWIFT outputs, not limited to but including: `yt` (Turk et al., 2011), `pynbody` (Pontzen, Roškar, Stinson, & Woods, 2013) and `pnbody` (Revaz, 2013). The other option is simply to use the `h5py` library directly, and forgo any extra processing of the data.

`swiftsimio` was created because these libraries either are too slow (usually performing significant pre-calculation when loading a snapshot), provide too weak of a connection to the data in the snapshot (e.g. reading the data in a different order than it is stored in the file), or would have been unable to integrate with the metadata that SWIFT outputs. To make full use of the SWIFT metadata all of these packages would have had to have significant re-writes of their internals, which would have taken significantly more time (due to their more complex codebases) or would have been unwelcome changes due to their impact on the users of other simulation codes.

We can also ensure that `swiftsimio` remains updated and constantly in-sync with the rapidly changing SWIFT code, as well as ensure that extra routines that are part of the library (e.g. visualisation) use the same definitions and functions that are implemented in the main simulation code.

Finally, `swiftsimio` includes many other features useful to users of the SWIFT cosmological simulation code, such as parallel visualisation and data repackaging.

The `swiftsimio` package will enable the next generation of cosmological simulations, ran with SWIFT, to be analysed on substantially smaller machines than were previously required with little extra effort from day-to-day users.

`swiftsimio` is hosted on GitHub and has documentation available through ReadTheDocs.

Acknowledgements

JB is supported by STFC studentship ST/R504725/1. AB is supported by STFC grants ST/R001006/1 and ST/P002447/1 as part of the DiRAC RSE team. This work used the DiRAC@Durham facility managed by the Institute for Computational Cosmology on behalf of the STFC DiRAC HPC Facility (www.dirac.ac.uk). The equipment was funded by BEIS capital funding via STFC capital grants ST/K00042X/1, ST/P002293/1, ST/R002371/1 and ST/S002502/1, Durham University and STFC operations grant ST/R000832/1. DiRAC is part of the National e-Infrastructure. We would like to extend our thanks specifically to Alastair Basden and his team for managing the DiRAC Memory Intensive service.

References

Borrow, J., Bower, R. G., Draper, P. W., Gonnet, P., & Schaller, M. (2018). SWIFT: Maintaining weak-scalability with a dynamic range of 10^4 in time-step size to harness extreme adaptivity. *Proceedings of the 13th SPHERIC International Workshop, Galway, Ireland, June 26-28 2018*, 44–51. Retrieved from https://ui.adsabs.harvard.edu/abs/2018arXiv180701341B/abstract

Goldbaum, N., ZuHone, J., Turk, M., Kowalik, K., & Rosen, A. (2018). Unyt: Handle, manipulate, and convert data with units in Python. *Journal of Open Source Software*, *3*(28), 809. doi:10.21105/joss.00809

Pontzen, A., Roškar, R., Stinson, G., & Woods, R. (2013, May). pynbody: N-Body/SPH analysis for python. http://ascl.net/1305.002.

Revaz, Y. (2013, February). pNbody: A python parallelized N-body reduction toolbox. http://ascl.net/1302.004.

Schaller, M., Gonnet, P., Chalk, A. B. G., & Draper, P. W. (2016). SWIFT: Using task-based parallelism, fully asynchronous communication, and graph partition-based domain decomposition for strong scaling on more than 100,000 cores. *Proceedings of the Platform for Advanced Scientific Computing Conference on - PASC '16*, 1–10. doi:10.1145/2929908.2929916

Schaye, J., Crain, R. A., Bower, R. G., Furlong, M., Schaller, M., Theuns, T., Dalla Vecchia, C., et al. (2015). The EAGLE project: Simulating the evolution and assembly of galaxies and their environments. *Monthly Notices of the Royal Astronomical Society*, *446*(1), 521–554. doi:10.1093/mnras/stu2058

Springel, V. (2005). The cosmological simulation code gadget-2. *Monthly Notices of the Royal Astronomical Society*, *364*(4), 1105–1134. doi:10.1111/j.1365-2966.2005.09655.x

The HDF Group. (1997). Hierarchical Data Format, version 5.

Turk, M. J., Smith, B. D., Oishi, J. S., Skory, S., Skillman, S. W., Abel, T., & Norman, M. L. (2011). yt: A Multi-code Analysis Toolkit for Astrophysical Simulation Data. *The Astrophysical Journal Supplement Series*, *192*(1), 9. doi:10.1088/0067-0049/192/1/9